

Description of SAM's CSP User-defined Power Cycle Model

DRAFT, Ty Neises, NREL Fall 2015

Motivation

SAM models parabolic trough, linear Fresnel, and molten salt power tower configurations that employ a heat transfer fluid (HTF) to absorb solar irradiance and deliver it as thermal power to a thermodynamic power cycle that utilizes steam as the working fluid. This type of configuration is known as an "indirect HTF" configuration, as opposed to "direct HTF" configurations wherein the power cycle working fluid also passes through the receiver (e.g. direct steam power tower). SAM's default indirect HTF power cycle model is a regression model developed from a detailed first-principles basis Rankine cycle model. This basis model calculates cycle performance over the expected cycle operating range by modeling each cycle component at off-design conditions. The model assumes that deviation in cycle performance at off-design conditions is independent of cycle design and only a function of deviation from the user specified design point. This model generally has been a fast, flexible, and accurate tool for most conventional CSP power cycles. However, some users have requested the capability to model their own Rankine cycle design or to model newer concepts that pursue the aggressive SunShot targets.

Approach

Overview

NREL has developed a user-defined power cycle option for SAM's indirect HTF technology models to meet this growing demand to model diverse and custom cycles. This option presupposes that the user has a custom power cycle model that can be used to generate cycle performance results over expected operating conditions. The methodology uses a structured design-of-experiments approach to guide and limit the number of custom power cycle simulations required. SAM provides data tables in its User Interface to store the user's performance data. SAM uses this tabular data to build a power cycle regression model that considers single variable effects and two variable interactions. The following sections explain the user-defined power cycle option in more detail.

Custom Power Cycle Model Requirements

SAM's indirect HTF component models use first-principles relationships to model the interaction between physical component design (e.g. receiver tube diameter, absorptivity, etc.), ambient conditions, and plant performance. Consequently, SAM's component models for the receiver, storage, and power cycle must conserve mass and energy as well as track the HTF temperature as the HTF passes between components. In order to integrate custom power cycle data into the existing indirect HTF technology models, the custom power cycle model must accept as inputs the HTF temperature ($T_{HTF,hot}$) and normalized mass flow rate (\dot{m}). Ambient temperature (T_{amb}) also influences the performance of thermodynamics power cycles and is the third independent input required of the custom model. Conceptually, the custom model calculates the outputs in the form of Equation (1), where Y represents any model output (e.g. cycle electric power generated). Because the technology models depend on the relationship between temperatures, mass flow rate, and thermal power, it is crucial that the custom cycle model is assuming HTF properties corresponding to the HTF selected in the SAM user interface (UI).

$$Y = f(\dot{m}, T_{HTF}, T_{amb}) \quad (1)$$

The custom model must return calculated metrics that define the cycle's performance; at a minimum, SAM requires the thermal power delivered to the cycle from the HTF (\dot{q}_{HTF}) and cycle electric power generated (\dot{W}_{cycle}). Given these values, SAM applies Equation (2) to calculate the HTF cold temperature returning to the receiver and/or thermal energy storage ($T_{HTF,cold}$), where c_p is the HTF specific heat at the average of the hot and cold temperatures. SAM also allows the user to optionally report calculated cooling parasitic load ($\dot{W}_{cooling}$) and cycle water use (\dot{m}_{water}). Because the cooling parasitic load is optional, the user must be sure that it is consistent with the reported cycle electric power generated. Equation (3) shows the relationship between the cycle net power calculated in SAM's regression model and the values reported from the user's custom model.

$$T_{HTF,cold} = T_{HTF,hot} - \frac{\dot{q}_{HTF}}{\dot{m} * c_p} \quad (2)$$

$$\dot{W}_{net,calculated} = \dot{W}_{cycle,custom} - \dot{W}_{cooling,custom} \quad (3)$$

Custom Cycle Design Point Performance

Because the custom power cycle model must interface with the CSP system defined in SAM, it is important to maintain consistency between the design points in SAM and the inputs and response of the custom cycle model. The SAM user interface contains user-specified and calculated inputs that define the custom cycle inputs and outputs at design. That is, two of the cycle inputs at design, HTF hot temperature and ambient temperature, are defined on the user interface, while the normalized mass flow rate is defined as 1.0 by convention. Similarly, the design point cycle efficiency, electric power generation, cooling parasitic load, and cycle water use are also defined in the user interface. Consequently, when solved with the design inputs, the custom power cycle outputs should match the corresponding values in SAM.

SAM's regression model requires that the user report the custom cycle model outputs normalized relative to their design values. As such, the normalized outputs will equal 1.0 at the design case.

Sampling the Custom Power Cycle Model

With a custom cycle model meeting the above requirements, the user must populate SAM's data tables with cycle outputs. The goal of the data tables is to accurately capture the custom cycle performance over practical ranges for each of the three independent inputs (for example, the normalized HTF mass flow rate may float between 0.3 and 1.1 during an annual simulation). One way to ensure that the tables represent the custom cycle over its expected operating conditions is to require the user to sample a dense mesh of input combinations. For example, if the user determines that 20 values accurately represent the range of possible values for each input, then the user would need to complete 8000 (i.e. 20^3) custom power cycle simulations. For many detailed process simulation software packages, this is a significant computational burden. Moreover, SAM would need to import all of the calculated data, and the regression model would need to expansively search through the data to find the correct interpolation region at any given set of inputs.

To reduce the computational requirements, SAM uses a multi-level design-of-experiments approach to limit the number of simulations required to represent the custom power cycle model by modeling single variable effects and two variable interactions. This approach requires that the user define *low* and *high* level values for each input, designated in Table 1 by – and + superscripts, respectively. The *low* level value should be less than the input's design value (designated by the * superscript) and greater than or equal to the lowest value of the input's practical range. For example, if the practical range of the normalized HTF

mass flow rate is from 0.3 and 1.1 and the design value is 1.0, then the *low* level of HTF mass flow rate could be 0.5 or 0.3, but not 0.2. Similarly, the *high* level value should be greater than the input's design value and less than or equal to the highest value of the input's practical range.

This approach requires nine parametric simulations of the custom cycle model: three for each input. First, the single variable (or *main*) effects are captured by a parametric analysis of the custom power cycle model over the practical range of the respective *main* input with the remaining two inputs at their design values, as shown by Parametric Analyses 2, 5, and 8 in Table 1. Next the *interaction* input for each *main* input is set to its *low* level, and the parametric analyses are rerun, as shown by Parametric Analyses 1, 4, and 7. Finally, the *interaction* inputs are set to their *high* levels, and the process repeated, as shown by Parametric Analyses 3, 6, and 9. In this way, the interaction is captured for each of the three possible combinations of two independent inputs. If the user selects 20 values to cover the practical range for each independent input, for example, then the approach outlined in Table 1 requires only 180 (i.e. 20×9) custom power cycle simulations.

Table 1: Custom Power Cycle Simulations Required to Populate SAM's Data Tables

SAM table	Parametric Analysis #	Number of Simulations	Custom Model Inputs		
			HTF Hot Temp	HTF Mass Flow Rate	Ambient Temperature
Table 1	1	$N_{T_{HTF,hot}}$	$T_{HTF,hot}^i$ for $i = 1..N_{T_{HTF,hot}}$	\dot{m}^-	T_{amb}^*
	2			\dot{m}^*	
	3			\dot{m}^+	
Table 2	4	$N_{\dot{m}}$	$T_{HTF,hot}^*$	\dot{m}^i for $i = 1..N_{\dot{m}}$	T_{amb}^-
	5				T_{amb}^*
	6				T_{amb}^+
Table 3	7	$N_{T_{HTF,amb}}$	$T_{HTF,hot}^-$	\dot{m}^*	$T_{HTF,hot}^i$ for $i = 1..N_{T_{HTF,hot}}$
	8		$T_{HTF,hot}^*$		
	9		$T_{HTF,hot}^+$		

Populating Data Tables in SAM

The user can simulate a custom power cycle model at the conditions in Table 1 to create data that SAM uses in its power cycle regression model. Next, the user must import that data to the data tables in the SAM User Interface. Rather than a unique table for each of the nine parametric analyses listed in Table 1, SAM groups into tables the parametric analyses that were calculated with the same *main* inputs. For example, analyses 1-3 were all calculated over the practical range of HTF hot temperatures. Consequently, the first column in the corresponding data table in SAM contains the HTF hot temperature, with the number of rows matching the number of values in the HTF hot temperature range. Then, the table provides *three consecutive columns for each calculated output: one column for each level of the interaction input*.

Figure 1 shows an example of how data is applied from the parametric analyses in Table 1 to the SAM data table containing results from parametric analyses of the practical range for the HTF hot temperature. Note that the *low* and *high* levels for the *interaction* input, \dot{m} , are

user inputs above the table. The data table containing the HTF mass flow rate results uses parametric analyses 4-6, and the table containing the ambient temperature results uses parametric analyses 7-9.

Low, design, and high mass flow rates (\dot{m}) for parameter interactions with HTF temperature:

Low normalized HTF \dot{m}

Design normalized HTF \dot{m}

High normalized HTF \dot{m}

Import... Export... Copy Paste Rows:

HTF temperature °C (at HTF $\dot{m} \Rightarrow$)	W cycle low	W cycle design	W cycle high	Heat in low	Heat in design	Heat in high	W cooling low	W cooling design	W cooling high	\dot{m} water low	\dot{m} water design	\dot{m} water high
$T_{HTF,hot}^{i=1}$	Parametric #1	Parametric #2	Parametric #3	Parametric #1	Parametric #2	Parametric #3	Parametric #1	Parametric #2	Parametric #3	Parametric #1	Parametric #2	Parametric #3
\vdots												
$T_{HTF,hot}^{i=N_{T_{HTF,hot}}}$												

Figure 1: Populating the HTF Temperature data table

SAM's Regression Model

SAM uses the normalized performance data that the user enters in the data tables to calculate cycle performance by fitting the data to the regression model in Equation (4). The three inputs, represented here by $T_{HTF,hot}$, \dot{m} , and T_{amb} , are passed to the regression model from the other CSP component models, and as such should fall within but not directly coincide with the *main* inputs in the respective data tables. Equation (4) is solved for each of the four outputs, and its normalized output is multiplied to the corresponding design value to calculate the output's absolute value.

$$\begin{aligned}
 Y = 1 + f_{ME,T_{HTF,hot}}(T_{HTF,hot}) + f_{ME,\dot{m}}(\dot{m}) + f_{ME,T_{amb}}(T_{amb}) + \\
 f_{INT,\dot{m} \rightarrow T_{HTF,hot}}^{\pm}(T_{HTF,hot}) * \frac{(\dot{m} - \dot{m}^*)}{(\dot{m}^* - \dot{m}^{\pm})} + f_{INT,T_{amb} \rightarrow \dot{m}}^{\pm}(\dot{m}) * \frac{(T_{amb} - T_{amb}^*)}{(T_{amb}^* - T_{amb}^{\pm})} \\
 + f_{INT,T_{HTF,hot} \rightarrow T_{amb}}^{\pm}(T_{amb}) * \frac{(T_{HTF,hot} - T_{HTF,hot}^*)}{(T_{HTF,hot}^* - T_{HTF,hot}^{\pm})}
 \end{aligned} \quad (4)$$

where:

- the $f_{ME,i}(i)$ terms represent the main effect of input i , linearly interpolated from the corresponding lookup table at i and the design value of the *interaction* input.
- The superscript \pm refers to either the lower or upper level of the *interaction* input, depending on whether the *interaction* input is less or greater than its design value, respectively.
- the $f_{INT,j \rightarrow i}^{\pm}(i)$ terms represent the interaction effect of input j on input i and are calculated two times for each input (one for the upper and one for the lower level of the *interaction* input) from the data tables at the beginning of the a simulation for each value in the practical range of i using Equation (5). When Equation (4) is applied during the annual CSP system simulation, these terms are calculated by linearly interpolating at i .

$$f_{INT,j \rightarrow i}^{\pm}(i) = -\left(Table_i(i, j^{\pm}) - 1.0 - f_{ME,j}(j^{\pm}) - f_{ME,i}(i) \right) \quad (5)$$

Summary

The following steps define the high-level process to successfully run the user-defined power cycle option in SAM.

1. Develop a custom power cycle model that accepts as inputs the HTF hot temperature, the normalized HTF mass flow rate (with respect to the design point mass flow rate), and the ambient temperature. Ensure that when applying the design point inputs, the calculated outputs match the corresponding values in SAM.
2. For each of the three model inputs:
 - a. Select a practical range covering expected cycle operating conditions over the course of the annual simulation. Create a sample of values within this range to accurately capture the cycle response over the operating range (i.e. select the number of values in the range).
 - b. Select *low* and *high* levels required when the input is the *interaction* input.
3. Complete the parametric analyses outlined in Table 1.
4. Using Figure 1 as a guide, populate the data tables in SAM using the normalized custom cycle results from the parametric analyses.
5. Run the SAM simulation.
6. Repeat these steps if you modify SAM inputs that affect the custom model results (e.g. the HTF temperature at design is increased).

Example

This section demonstrates the procedure listed above using a very simple custom power cycle model. The code snippets below are in *Python*.

1. First, we developed a class in Python to model the simple custom power cycle and title it 'generic_power_cycle'. The first method in the class is the required '__init__' call that *Python* uses when an instance of the class is declared. We defined this method to take as inputs the cycle design inputs: HTF cold return temperature, thermal efficiency, thermal power, HTF hot temperature, ambient temperature, and normalized mass flow rate. The method then calculates the design outputs: cycle electric power generation, the endoreversible thermal efficiency, and the specific heat required to fulfill the energy balance. The second method in this class is 'off_design', and it calculates the cycle performance given the inputs: HTF hot temperature, normalized HTF mass flow rate, and ambient temperature. Note that this method takes the form of Equation (1). To simplify the model, the cycle performance calculations assume that the HTF cold return temperature is always equal to its design value. Additionally, the simplified model assumes that the cycle cooling parasitic load and water use is negligible.

```
class generic_power_cycle:
```

```
    def __init__(self, T_htf_return, eta_ref, Q_dot_ref, T_htf_ref, T_amb_ref, m_dot_ref):
        self.T_htf_return = T_htf_return
        self.eta_ref = eta_ref
        self.Q_dot_ref = Q_dot_ref
        self.T_htf_ref = T_htf_ref
        self.T_amb_ref = T_amb_ref
```

```

self.m_dot_ref = m_dot_ref
self.W_dot_ref = self.Q_dot_ref*self.eta_ref
self.cp = self.Q_dot_ref/(self.m_dot_ref*(self.T_htf_ref - self.T_htf_return))
self.eta_endorev_ref = 1.0 -
np.sqrt((self.T_amb_ref+273.15)/(self.T_htf_ref+273.15))

def off_design(self, T_htf_od, T_amb_od, m_dot_od):
    Q_dot_od = m_dot_od*self.cp*(T_htf_od - self.T_htf_return)
    eta_endorev_od = 1.0 - np.sqrt((T_amb_od+273.15)/(T_htf_od+273.15))
    eta_od = ((self.m_dot_ref - abs(self.m_dot_ref-
m_dot_od))/self.m_dot_ref)**0.3*(eta_endorev_od/self.eta_endorev_ref)*self.eta_ref
    W_dot_od = eta_od*Q_dot_od
    self.Q_dot_od_ND = Q_dot_od / self.Q_dot_ref
    self.W_dot_od_ND = W_dot_od / self.W_dot_ref
    self.W_dot_cool_ND = 1.0
    self.water_use_ND = 1.0

```

Next, we added code to open files that will contain values for the three data tables in SAM and name the columns that the code will be populating:

```

T_htf_csv = open("T_htf.csv","w")
T_amb_csv = open("T_amb.csv","w")
m_dot_htf_csv = open("m_dot_htf.csv","w")

common_hdrs = ["W_dot_low", "W_dot_ref", "W_dot_high", "Q_dot_low", "Q_dot_ref",
"Q_dot_high", "W_dot_cool_low", "W_dot_cool_ref", "W_dot_cool_high", "water_low",
"water_ref", "water_high"]

T_htf_hdrs = ["T_htf"] + common_hdrs
T_amb_hdrs = ["T_amb"] + common_hdrs
m_dot_htf_hdrs = ["m_dot_htf"] + common_hdrs

```

Finally, to complete this step, we set the required cycle design values using the default values from SAM's molten salt power tower model:

```

"Set up design parameters"
T_htf_return = 290;    #C
eta_ref = 0.412;      #C
Q_dot_ref = 279.1;    #MW

```

"Independent variables at design"

```
T_htf_ref = 574.0;    #C
```

```
T_amb_ref = 43.0;    #C
```

```
m_dot_htf_ref = 1.0; #-
```

"Initialize generic power cycle class"

```
c_PC = generic_power_cycle(T_htf_ref, eta_ref, Q_dot_ref, T_htf_ref, T_amb_ref,
m_dot_htf_ref);
```

2. Next, we selected the upper and lower values for the practical range of each input. We also set the *low* and *high* level values to these limits, *although they are not required to be equal*. We chose to have 20 values in the practical range for each value and will calculate them equidistantly in the next step. The remaining lines of code prepare lists to handle the *low*, *design* and *high* level for each input as well as the calculated input value over its practical range.

"Low and High level values of independent variables"

```
T_htf_low = 500.0    #C
```

```
T_htf_high = 580.0   #C
```

```
T_amb_low = 0.0;     #C
```

```
T_amb_high = 55.0;   #C
```

```
m_dot_htf_low = 0.3;  #-
```

```
m_dot_htf_high = 1.2;  #-
```

```
N_runs = 20;
```

```
T_htf_levels = [T_htf_low, T_htf_ref, T_htf_high]
```

```
T_amb_levels = [T_amb_low, T_amb_ref, T_amb_high]
```

```
m_dot_htf_levels = [m_dot_htf_low, m_dot_htf_ref, m_dot_htf_high]
```

```
T_htf_parametric = {}
```

```
T_amb_parametric = {}
```

```
m_dot_htf_parametric = {}
```

```
for key in T_htf_hdrs:
```

```
    T_htf_parametric[key] = [None]*N_runs
```

```
for key in T_amb_hdrs:
```

```
    T_amb_parametric[key] = [None]*N_runs
```

```
for key in m_dot_htf_hdrs:
```

```
    m_dot_htf_parametric[key] = [None]*N_runs
```

3. Then we set up a *for* loop for each of the 20 values in the input practical ranges (using the same number of values in each range simplifies the required code). Next we added an inner *for* loop for the three levels of the *interaction* input. Using these loops, we calculated the correct inputs to the 'off_design' method and fill the text files for each of the data tables in SAM.

```

for i in range(N_runs):

    for j in range(len(T_htf_levels)):

        "First, set parametric values for each independent variable"
        if( j == 0 ):
            T_htf_parametric["T_htf"][i] = T_htf_low + (T_htf_high-T_htf_low)/float(N_runs-1)*i
            T_amb_parametric["T_amb"][i] = T_amb_low + (T_amb_high-T_amb_low)/float(N_runs-1)*i
            m_dot_htf_parametric["m_dot_htf"][i] = m_dot_htf_low + (m_dot_htf_high-m_dot_htf_low)/float(N_runs-1)*i

        "Set off design values for T_htf parametric"
        T_htf_od = T_htf_parametric["T_htf"][i]
        m_dot_htf_od = m_dot_htf_levels[j]
        T_amb_od = T_amb_ref
        c_PC.off_design(T_htf_od, T_amb_od, m_dot_htf_od)
        T_htf_parametric[str(common_hdrs[0+j])][i] = c_PC.W_dot_od_ND
        T_htf_parametric[str(common_hdrs[3+j])][i] = c_PC.Q_dot_od_ND
        T_htf_parametric[str(common_hdrs[6+j])][i] = c_PC.W_dot_cool_ND
        T_htf_parametric[str(common_hdrs[9+j])][i] = c_PC.water_use_ND

        "Set off design values for T_amb parametric"
        T_amb_od = T_amb_parametric["T_amb"][i]
        T_htf_od = T_htf_levels[j]
        m_dot_htf_od = m_dot_htf_ref
        c_PC.off_design(T_htf_od, T_amb_od, m_dot_htf_od)
        T_amb_parametric[str(common_hdrs[0+j])][i] = c_PC.W_dot_od_ND
        T_amb_parametric[str(common_hdrs[3+j])][i] = c_PC.Q_dot_od_ND
        T_amb_parametric[str(common_hdrs[6+j])][i] = c_PC.W_dot_cool_ND
        T_amb_parametric[str(common_hdrs[9+j])][i] = c_PC.water_use_ND

```

```

"Set off design values for m_dot_htf parametric"
m_dot_htf_od = m_dot_htf_parametric["m_dot_htf"][i]
T_amb_od = T_amb_levels[j]
T_htf_od = T_htf_ref
c_PC.off_design(T_htf_od, T_amb_od, m_dot_htf_od)
m_dot_htf_parametric[str(common_hdrs[0+j])][i] = c_PC.W_dot_od_ND
m_dot_htf_parametric[str(common_hdrs[3+j])][i] = c_PC.Q_dot_od_ND
m_dot_htf_parametric[str(common_hdrs[6+j])][i] = c_PC.W_dot_cool_ND
m_dot_htf_parametric[str(common_hdrs[9+j])][i] = c_PC.water_use_ND

for j in range(len(common_hdrs)+1):
    add_end = ","
    if( j == len(common_hdrs) ):
        add_end = "\n"
    T_htf_csv.write(str(T_htf_parametric[str(T_htf_hdrs[j])][i])+add_end)
    T_amb_csv.write(str(T_amb_parametric[str(T_amb_hdrs[j])][i])+add_end)
m_dot_htf_csv.write(str(m_dot_htf_parametric[str(m_dot_htf_hdrs[j])][i])+add_end)

T_htf_csv.close()
T_amb_csv.close()
m_dot_htf_csv.close()

```

4. The previous steps shows code that generates text files containing data that is formatted such that the file can be directly imported to the data tables in SAM. We used the *Import...* button above each data table to load the corresponding text file.
5. We simulated the user-defined power cycle option in SAM using the imported data generated by the *Python* code. The annual energy generation is about 1% greater than the results from the default power cycle model. This comparison helps show that the regression model in SAM is correctly modeling the information in the data tables.